Data streaming, workflow and firewall-friendly steerable Grid Services with Styx

Jon Blower, Keith Haines
Reading e-Science Centre
University of Reading

Ed Llewellin
BP Institute for Multiphase Flow
University of Cambridge

Motivation

- Transfers of large amounts of binary data in Web Service-based workflows are problematic
  - Inefficient to encode data in XML, even as attachments
  - Data shouldn't follow same path as SOAP messages: shouldn't go through workflow engine
  - Large problem in environmental e-Science (e.g. GODIVA)
- Asynchronous messaging often hard to implement reliably
  - Firewalls and NAT can block incoming messages
- General need for lightweight middleware
  - Even if it doesn't support entire possible feature set

Solution: the Styx Grid Service (SGS)

- Wraps an unmodified binary executable and provides access to its streams (stdin, stdout, stderr) over the Internet
- Data can be streamed directly from service to service in workflows (like pipes on Unix command line)
- Software is lightweight with few dependencies
  - Easy to install
- Clients require no incoming ports to be open
- Designed with usability high on agenda
  - Responsive GUIs, easy deployment
- See http://jstyx.sf.net and paper accompanying this talk for more details

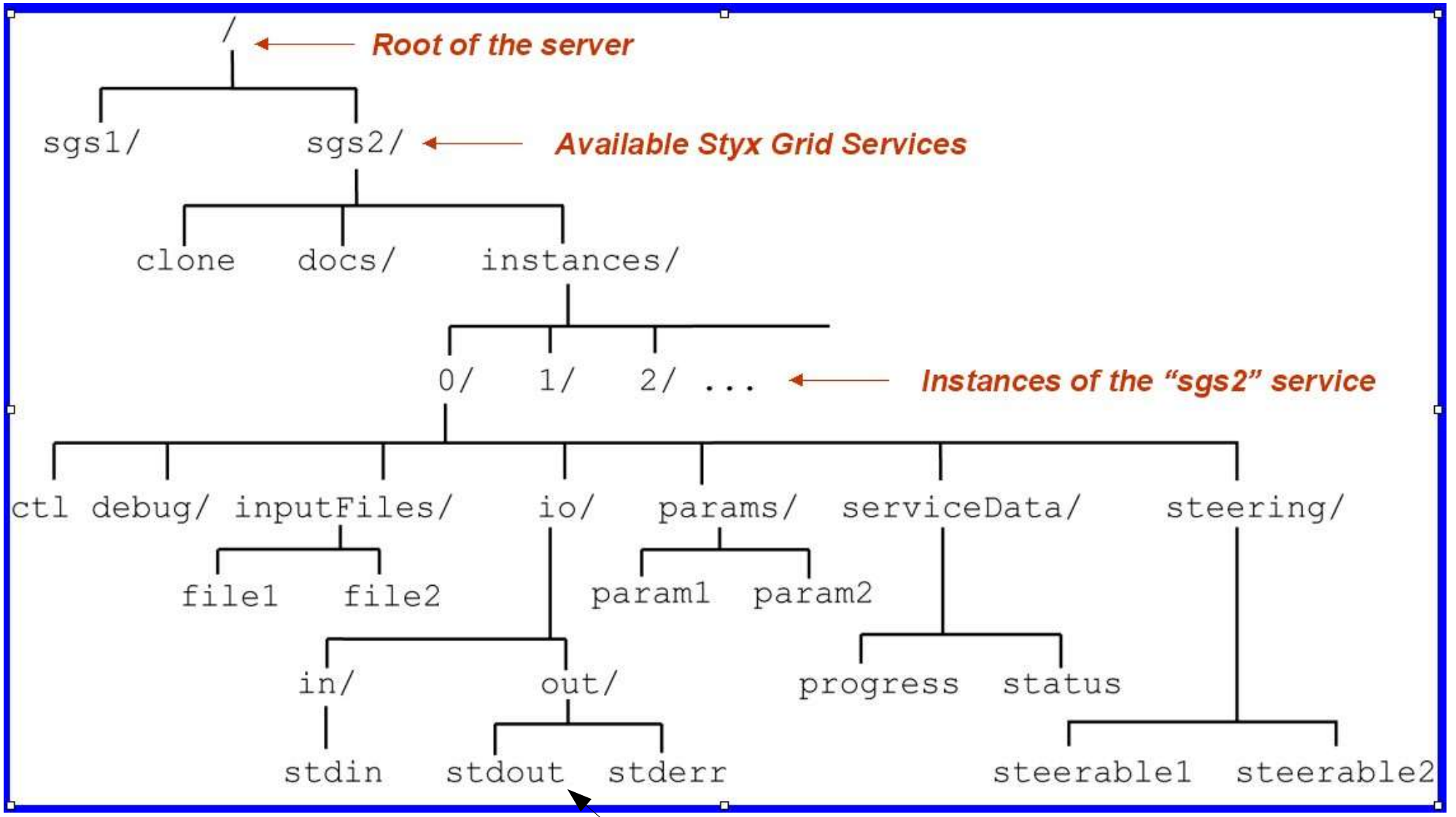
The Styx protocol for distributed systems

- Well-established (~20 years old, Bell Labs, Pike & Ritchie)
  - Core of Inferno and Plan9 operating systems (Styx = 9P2000)
- Virtualizes all resources as files
  - Files, blocks of RAM, devices, databases etc. all represented as a file or a set of files known as a *namespace*
  - Only needs a small command set (only has to deal with files). 13 commands in total (open, read, write, close, delete etc)
  - Therefore it is very lightweight (small protocol overhead)
- Each resource can be represented by a URL
- "Accidentally RESTful"
- Typically use *persistent connections* (it is a stateful protocol)
  - Hence clients don't need any *inbound* ports open through firewalls and NAT is no problem

Styx Grid Service basics

- Based on the Styx protocol
  - Using JStyx libraries (pure-Java Styx implementation)
- Key features:
  - Access to streams (stdin, stdout, stderr): can get live data
  - Contains asynchronous messaging system for monitoring progress, status data etc (no need for incoming firewall holes)
  - Supports computational steering (adjustment of parameters in-running)
- Clients make connection to SGS server and interact through reading and writing the virtual files
- Every "file" can be represented as a URL (pointer to a resource: cf. Grid Service Handle, WS-Notification Topic etc)

Namespace of an SGS server



styx://server:port/sgs2/instances/0/io/out/stdout

Data streaming

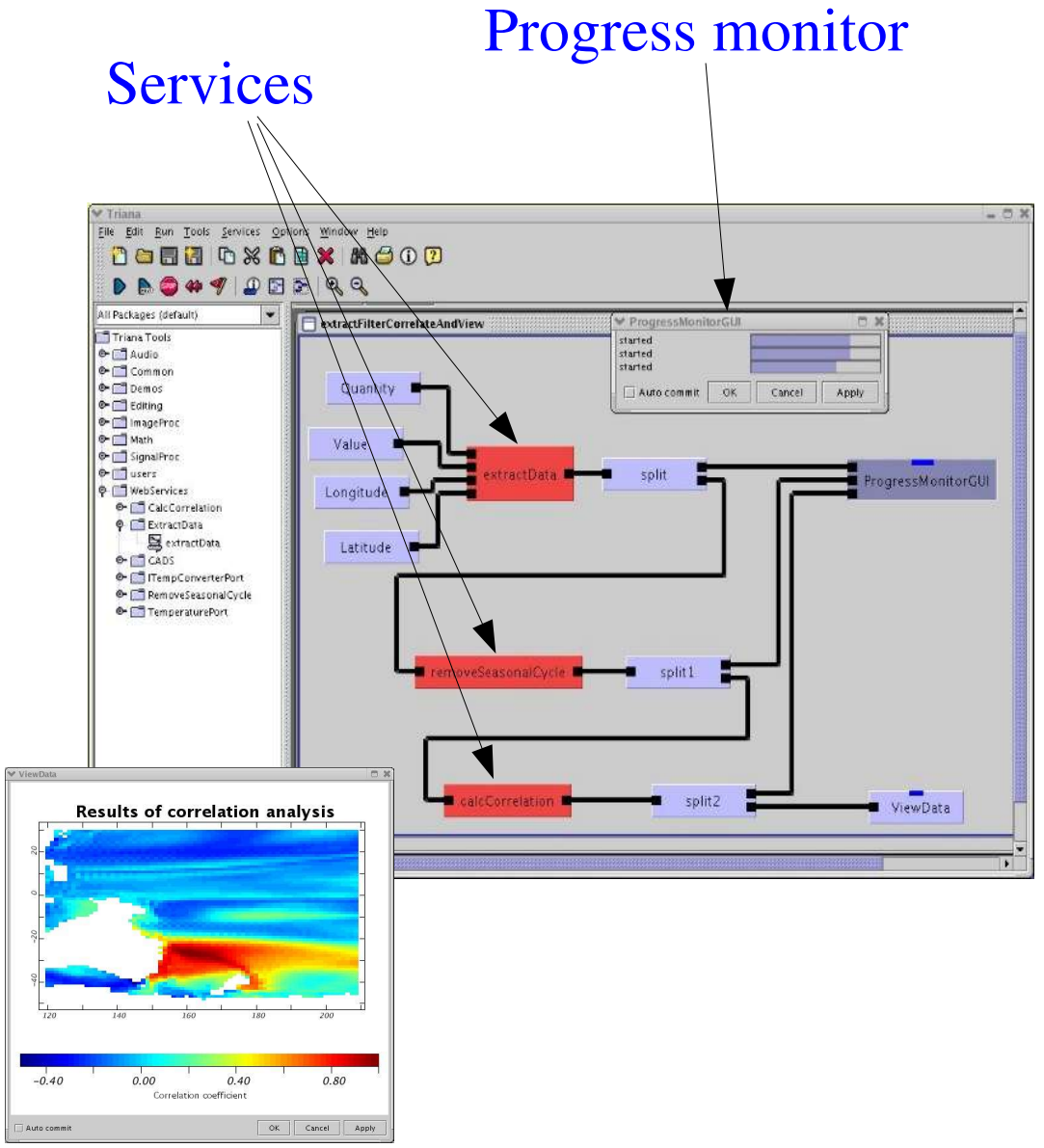
- Definition of a "stream" is data that is produced *while a program is running*
  - e.g. stdout, stderr, or any other output files
  - (Contrast with output files that are only available once a program has completed.)
- In Styx, reading from a stream is the same as reading from any other file
  - Clients make requests for data from a file, and receive the data as soon as it is available
- In SGS, any number of clients can read from the same file at the same time

SGS and workflow

- SGS solves the problem of transferring large amounts of binary data in workflows
- Data can be streamed directly from service to service
  - Workflow enactors pass around pointers to streams, i.e. the URLs of the stream files in the SGS namespace
- Closely analogous to using pipes on the Unix command line "**`prog1 | prog2 | prog3`**"
- Built into latest version of Taverna
- Means that services can execute *concurrently*
  - "prog2" can process some data while "prog1" is still running, and so forth
  - Increases efficiency of workflows

Workflow case study

- Correlation analysis on large ocean data set
- Three Services:
  - Extract data
  - Filter out seasonal cycle
  - Calculate correlation map
- Wrapped SGSs as Web Services, then executed workflow in Triana
- Control flow = SOAP, data flow = Styx
- Triana monitors progress via Styx interface



Asynchronous messaging

- JStyx libraries make it very easy to create messaging frameworks
- E.g. topic-based publish-subscribe ("pub-sub")
  - Each topic is a virtual file
  - Clients write to this file to update topic ("publish")
  - Clients read from this file ("subscribe"): when the contents change, they get new data immediately
- Messages very small (very little overhead to actual message payload – few bytes per message)

Collaborative visualization

- Any number of clients can read from a live output stream from an SGS instance at the same time
- Hence a number of clients can visualize the same data at the same time
- Clients pull data from the server (the server doesn't push)
  - So the server doesn't care how many clients there are

Computational steering

- "Computational steering" is when the parameters of a program can be altered while the program is running
- Usually used in conjunction with live (perhaps collaborative) visualization
- Very useful for exploring parameter space
  - Just like twiddling the knobs on a piece of lab apparatus
  - Encourages "playing"!
  - Also can quickly find problems in model code
- Easiest way to implement is for the program to periodically read parameters from local files on the server
  - In SGS, these files are edited over the Internet via Styx

Demo

Live collaborative visualization and computational steering of Lattice Boltzmann program

Notes about demo

- Ed wrote program that outputs data on stdout
  - I wrapped as SGS (5 mins. Work)
  - Ed didn't need to know anything about Styx Grid Services
- Just one server program and one port at the server side
  - Same port handles all file transfer, messaging etc
  - Clients need no incoming ports open; just need to make outgoing connection to that one port on the server
- Asynchronous messaging: when one client updated steering parameter, others got notified
- Could have web interface
  - Set up another service to convert output data into JPEG

Future work

- SSL-based security – not too hard to implement, nearly there!
  - Certificates for authentication; Styx traffic will be encrypted
- Wrap SGS as WS-Resource (for WSRF compatibility)
  - Being investigated by Andrew Harrison at Cardiff
  - "Compatibility where necessary, performance where possible"
- Investigate collaboration with WEDS
  - RealityGrid steering system
- Further case studies
  - Improves features, make software more robust
- Look at http://jstyx.sf.net - join mailing lists!